

**IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF NEW YORK**

INTERNATIONAL BUSINESS
MACHINES CORPORATION,

Plaintiff

v.

PLATFORM SOLUTIONS, INC.,

Defendant

Civil Action No. 06 CV 13565 (LAK)

[REDACTED VERSION]

DEFENDANT PSI'S CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. LAW OF CLAIM CONSTRUCTION	6
III. FOUR DISPUTED CLAIM TERMS COMMON TO SEVERAL PATENTS.....	7
A. "Processor"	8
B. "Instruction"	18
C. "Register"	25
D. "Program Status Word"	32
IV. THE INSTRUCTION/ARCHITECTURE PATENTS.....	36
A. The '495 "Resume Program" Patent	36
B. The '789 "Time Stamp" Patent.....	55
V. THE EMULATION PATENTS	66
A. The '520 "Address Translation" Patent.....	66
B. The '261 "Patching" Patent	81
VI. THE FLOATING POINT PATENTS	92
A. The '106 "Binary Floating Point Using Hexadecimal Floating Point Unit" Patent	94
B. The '678 "Data Class" Patent.....	110
C. The '709 "Rounding Modes" Patent	126
VII. THE PARTITIONING PATENTS.....	127
A. The '812 "Partition Communication" Patent.....	126
B. The '002 "Firmware Booting" Patent.....	139
VIII. THE '851 I/O PATENT.....	152
IX. CONCLUSION	156

TABLE OF AUTHORITIES

Cases

<i>Alloc, Inc. v. ITC</i> , 342 F.3d 1361 (Fed. Cir. 2003)	84
<i>Aristocrat Techs. Austl. PTY Ltd. v. Int'l Game Tech.</i> , 521 F.3d 1328 (Fed. Cir. 2008)	passim
<i>Atmel Corp. v. Info. Storage Devices</i> , 198 F.3d 1374 (Fed. Cir. 1999)	78
<i>Biomedino, LLC v. Waters Techs. Corp.</i> , 490 F.3d 946 (Fed.Cir.2007)	44, 137
<i>Braun Med., Inc. v. Abbott Labs.</i> , 124 F.3d 1419 (Fed. Cir. 1997)	passim
<i>Data General Corp. v. IBM</i> , 94-cv-12213 (D. Mass.).....	5, 21, 25
<i>Default Proof Credit Card Sys. v. Home Depot U.S.A., Inc.</i> , 412 F.3d 1291 (Fed. Cir. 2005)	passim
<i>Epcon Gas Sys. v. Bauer Compressors, Inc.</i> , 279 F.3d 1022 (Fed. Cir. 2002)	89
<i>Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.</i> , 535 U.S. 722 (2002)	55
<i>Gillespie v. Dywidag Sys. Int'l</i> , 501 F.3d 1285 (Fed. Cir. 2007)	131
<i>Griffin v. Bertina</i> , 285 F.3d 1029 (Fed. Cir. 2002)	63
<i>Honeywell Int'l, Inc. v. ITT Industries, Inc.</i> , 452 F.3d 1312 (Fed. Cir. 2006)	153, 154
<i>Intergraph Hardware Technologies Co. v. Toshiba Corp.</i> , 508 F. Supp. 2d 752 (N.D. Cal. 2007).....	63
<i>Lockheed Martin Corp. v. Space Systems/Loral, Inc.</i> , 324 F.3d 1308 (Fed. Cir. 2003)	63
<i>Markman v. Westview Instruments, Inc.</i> , 52 F.3d 967 (Fed. Cir. 1995)	7

<i>Massachusetts Institute of Technology v. Abacus Software</i> , 462 F.3d 1344 (Fed Cir. 2006)	106
<i>Microsoft Corp. v. Multi-Tech Sys., Inc.</i> , 357 F.3d 1340 (Fed. Cir. 2004)	passim
<i>Omega Eng'g, Inc. v. Raytek Corp.</i> , 334 F.3d 1314 (Fed.Cir. 2003)	44
<i>On Demand Mach. Corp. v. Ingram Indus., Inc.</i> , 442 F.3d 1331 (Fed. Cir. 2006)	153
<i>SciMed Life Systems, Inc. v. Advanced Cardiovascular Systems, Inc.</i> , 242 F.3d 1337 (Fed. Cir. 2001)	154
<i>Seachange Int'l, Inc. v. C-Cor Inc.</i> , 413 F.3d 1361 (Fed. Cir. 2005)	58
<i>Sinorgchem Co. v. ITC</i> , 511 F.3d 1132 (Fed. Cir. 2007)	83
<i>Watts v. XL Sys., Inc.</i> , 232 F.3d 877 (Fed. Cir. 2000)	154

Statutes

35 U.S.C. § 112	passim
-----------------------	--------

I. INTRODUCTION

There are 10 patents and 79 disputed claim terms before the Court. Each of the patents relates to aspects of mainframe computer technology. For the Court's convenience, PSI has prepared a 1-page overview of the 10 patents, including their filing dates and a brief statement of the claimed inventions. This chart lists the patents in chronological order and is attached as Exhibit 1 to the Declaration of Tibor L. Nagy (hereinafter "Nagy Decl."), filed concurrently.

The principal dispute between the parties with respect to claim construction is whether the disputed terms should be defined amorphously and by their function alone (IBM's position) or whether the terms have more concrete meanings in the patents and to those of ordinary skill in the art (PSI's position). For example, the following six terms, which are among the most significant terms in dispute, emphasize the parties' respective contentions:

TERM	IBM'S CONSTRUCTION	PSI'S CONSTRUCTION
processor	<i>A portion of a computer system</i> that interprets and executes instructions.	<i>One or more integrated circuits</i> that process coded instructions and perform a task
floating point processor	<i>A portion of a computer system</i> that interprets and executes floating point instructions.	A <i>processor</i> that contains a <i>floating point unit</i> . A floating point unit is that portion of a processor's circuitry that executes floating point instructions by performing operations on floating point numbers.
floating point unit	<i>Part of a computer system</i> that performs floating point operations.	That <i>portion of a processor's circuitry</i> that performs floating point calculations.
converter	<i>Hardware and/or software</i> that changes data from one format or architecture type to another format or architecture type.	<i>Circuitry within the floating point unit</i> that changes the format or architecture type of a floating point number.
instruction	A <i>language construct</i> that specifies an operation and identifies its operands, if any.	A <i>string of digits</i> that specifies an operation and identifies its operands, if any, and <i>can be directly executed by the processor</i> to which it is directed.
register	A <i>part of internal storage</i> having a specified storage capacity and usually intended for a specific purpose.	A <i>hardware storage element in the processor</i> that can be accessed faster than memory.

As in all patent cases, the parties' constructions are motivated by non-infringement and invalidity arguments. Here, the non-infringement arguments are most transparent in the parties' respective constructions. IBM's proposed constructions are amorphous (e.g., "part of a computer system...", "a portion of a computer system...", "hardware and/or software...") for two reasons. First, as IBM's 30(b)(6) witness admitted at his deposition, [REDACTED] (Ex. 2).¹ Thus, IBM needs "processor," "floating point unit," and other terms that obviously refer to hardware to be construed amorphously, so that they can be read onto PSI's software.

Second, the processors in PSI's machines are all manufactured by Intel and have a different Instruction Set Architecture ("ISA") than the processors in IBM's machines. For this reason, the processors in the accused devices are *incapable* of executing the instructions that are claimed in several of the patents-in-suit. For example, it is undisputed that the Intel processors in all of the accused devices are *incapable* of executing the "Resume Program Instruction" of the '495 patent or the "Test FP Data Class Instruction" of the '678 patent. *See* IBM Br. at 7 (acknowledging that "Intel-based computers...are incapable of understanding and executing IBM instructions"). And that is why, among other things, IBM objects to PSI's elementary contention that an "instruction" can be "directly executed by the processor to which it is directed." Instead, IBM wants "instruction" to be construed awkwardly as "a language construct." Computers do not execute language constructs; they execute instructions.

Apart from being inconsistent with the intrinsic evidence, the problem with IBM's amorphous constructions is that they result in absurdities that no rational person of ordinary skill in the art would ever accept. For example, IBM's construction of "processor" is so broad that, as

¹ "Ex." refers to the exhibits attached to the Nagy Decl. *See* Ex. 2 at 273:7-25, 280:25-282:8, 295:19-296:8 [REDACTED]

acknowledged by the inventor of the '520 patent,² under this construction a processor includes, among other things, a keyboard and a monitor:

114

Table 1. Continued

² This inventor, Soumya Mallick, [REDACTED] (Ex. 3).

[REDACTED]

[REDACTED]

Similarly, IBM's construction of "floating point unit" also encompasses keyboards and monitors, and its construction for "converter" ("hardware and/or software that changes data from one format...to another format") is broad enough to include keyboards, monitors, modems, printers, Blackberries, cell phones, and just about any piece of modern digital technology.

In other contexts, when IBM is not trying to make its patents read on machines whose hardware is already licensed to all of IBM's patents, IBM does not use the disputed claim terms in such awkward ways. For example, PSI invites the Court to read pages 20-21 of IBM's 1999 Claim Construction Brief in *Data General Corp. v. IBM*, 94-cv-12213 (D. Mass.) (Ex. 4). IBM's discussion of "instructions" and "machine instructions" in that brief is completely inconsistent with the awkward "language construct" definition that IBM proposes here—and is substantively identical to PSI's proposed construction for "instruction." PSI also invites the Court to read the 1-page mainframe technology summary in U.S. Pat. No. 7,117,389 (the '389 patent) (Ex. 5), a patent not at issue in this lawsuit but which is assigned to IBM. That useful 1-page summary makes abundantly clear that, consistent with PSI's proposed constructions, a processor is "an integrated circuit on a single chip" and that "a floating point unit...requires a significant area of an integrated circuit chip." '389 patent at 2:1-2, 41-43. And by simply browsing IBM's website (www.ibm.com), the Court will find scores of articles that make IBM's proposed constructions look awkward and anachronistic.⁴ Notably, the only examples IBM's expert Dr. Smotherman could dig up to support IBM's construction for "processor" were from 1964 and 1971.

³ Mallick Depo. (Ex. 2) at 57:5-58:24.

⁴ The following useful article, for example, flatly contradicts what IBM tells the Court about "processors," "registers," and "floating point units": http://www.ibm.com/podcasts/howitworks/20080225/index.shtml?sa_campaign=message/ideas/leadspace/all/semiconductorsflash.

Smotherman ¶¶ 5-8. And the only example Dr. Smotherman could find to support IBM's constructions for "program status word" and "register" was the same single article from *1964. Id.* ¶¶ 13-15. Dr. Smotherman's report reads more like a museum catalog than a serious attempt at explaining the technology at issue in this lawsuit.

In the remainder of this brief, PSI discusses each of the 10 patents-in-suit and the intrinsic and extrinsic evidence relevant to the disputed claim terms. With respect to the particular technology at issue, PSI directs the Court to its Technology Tutorial CD (filed on June 2, 2008), and will not duplicate that material here. Finally, PSI notes that IBM devotes an entire section of its brief to rhetoric accusing PSI of "theft of IBM's patents, trade secrets, and copyrights," denigrating PSI's machines as "clones" and "knock-offs," and praising its own product development as "one of the three most important business innovations of all time." IBM Br. at 1, 4, 8. Because such rhetoric is irrelevant to claim construction, PSI will not respond to it here, other than to quote the following testimony from a neutral third-party consumer who, but for IBM's anticompetitive conduct, would be able to continue using the Liberty server he purchased from PSI for half the cost of IBM's artificially overpriced mainframes:

Q: How has that [IBM's z/OS operating system] performed on the Liberty server?

A: Flawlessly.

Q: How has the Liberty server performed overall?

A: Flawlessly.

Q: And you stated earlier that there is the additional functionality [on PSI's machine] of being able to run a Windows—Windows or other operating system based platform, is that correct?

A: Correct.

Q: Okay. Did you incur any problems in switching from the IBM machine to the Liberty server?

A: Zero.

Deposition of Ronald Avery (Ex. 6), testifying on behalf of Polk County, Iowa (an early PSI customer), at 45:11-46:15.

II. LAW OF CLAIM CONSTRUCTION

“It is a ‘bedrock principle’ of patent law that ‘the claims of a patent define the invention to which the patentee is entitled the right to exclude.’” *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc) (quoting *Innova/Pure Water Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1115 (Fed. Cir. 2004)). “[T]he words of a claim ‘are generally given their ordinary and customary meaning.’” *Id.* (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)). When “the meaning of a claim term as understood by persons of skill in the art” is not “immediately apparent, ... the court looks to ‘those sources available to the public that show what a person of skill in the art would have understood disputed claim language to mean.’” *Id.* at 1314 (quoting *Innova*, 381 F.3d at 1116). “Those sources include ‘the words of the claims themselves, the remainder of the specification, the prosecution history, and extrinsic evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art.’” *Id.* at 1314 (quoting *Innova*, 381 F.3d at 1116).

“Intrinsic evidence is the most significant source of the legally operative meaning of disputed claim language.” *Vitronics*, 90 F.3d at 1582. When reviewing intrinsic evidence, the Court first “look[s] to the words of the claims themselves, both asserted and nonasserted,” then “review[s] the specification to determine whether the inventor has used any terms in a manner inconsistent with their ordinary meaning,” and finally “consider[s] the prosecution history of the patent if in evidence.” *Id.* The Court may also rely on “extrinsic evidence, which ‘consists of all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises.’” *Phillips*, 415 F.3d at 1317 (quoting *Markman v. Westview*

Instruments, Inc., 52 F.3d 967, 980 (Fed. Cir. 1995) (en banc)). Such evidence, however, “is ‘less significant than the intrinsic record in determining the legally operative meaning of claim language.’” *Id.* (quoting *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 862 (Fed. Cir. 2004)). The Court has “especially noted the help that technical dictionaries may provide to a court ‘to better understand the underlying technology’ and the way in which one of skill in the art might use the claim terms.” *Id.* at 1318 (quoting *Vitronics*, 90 F.3d at 1584 n. 6). Additionally, “expert testimony can be useful to a court for a variety of purposes, such as to provide background on the technology at issues, to explain how an invention works, to ensure that the court’s understanding of the technical aspects of the patent is consistent with that of a person of skill in the art, or to establish that a particular term in the patent or prior art has a particular meaning in the pertinent field.” *Id.* at 1318.

Where a patent contains “means-plus-function” limitations, section 112, Paragraph 6 mandates that “such a claim limitation ‘be construed to cover the corresponding structure . . . described in the specification and equivalents thereof.’” *Braun Med., Inc. v. Abbott Labs.*, 124 F.3d 1419, 1424 (Fed. Cir. 1997) (quoting 35 U.S.C. § 112, ¶ 6). “[P]ursuant to this provision, structure disclosed in the specification is ‘corresponding’ structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim.” *Id.*; *Medtronic, Inc. v. Advanced Cardiovascular Sys.*, 248 F.3d 1303, 1311 (Fed. Cir. 2001).

III. FOUR DISPUTED CLAIMS TERMS COMMON TO SEVERAL PATENTS

Four of the disputed claim terms appear in several patents, and the parties have proposed competing constructions for these terms that apply to all of the patents. PSI will begin by addressing these four terms. PSI will then address the remaining terms on a patent-by-patent basis. The common terms are:

1. “processor” (‘495, ‘520, ‘261, ‘789, ‘678 and ‘709 patents)

2. “instruction” (‘495, ‘789, ‘261 and ‘709 patents)
3. “program status word” (‘495 and ‘678 patents)
4. “register” (‘495 and ‘789 patents)

A. “Processor”⁵

IBM’s Proposed Construction	PSI’s Proposed Construction
A portion of a computer system that interprets and executes instructions.	One or more integrated circuits that process coded instructions and perform a task.

The principal dispute between the parties is whether a “processor” is composed of one or more integrated circuits or whether it can be any “portion of a computer system,” including software programs. While failing to identify a single example of a processor that does not contain an integrated circuit, IBM defends its amorphous construction by arguing that PSI’s proposal excludes the possibility of “microcode” being part of a processor. That argument lacks merit because it mistakenly equates “microcode,” which PSI *agrees* is part of the processor, with “software” generally, which the intrinsic and extrinsic evidence overwhelmingly demonstrate is not.⁶ Though “microcode” is *similar* to software, in that it is composed of strings of digits (*i.e.*, 0s and 1s), the intrinsic evidence expressly teaches that microcode is ***built into*** the integrated circuits of a processor and is therefore a ***“manufactured component of the system.”*** ‘495 patent at 4:57-58 (emphasis added). PSI’s proposal should be adopted because it is supported by the intrinsic evidence and because IBM’s sole argument to the contrary is demonstrably false.

As an initial matter, and to simplify the analysis, PSI believes that there is no material difference in the parties’ proposed constructions after the word “that.” Specifically, IBM’s brief made clear that IBM is using “interprets” and “decodes” interchangeably. *See* IBM Br. at 14 (“a

⁵ The term “processor” is used in claims 4, 7 and 19 of the ‘495 patent; claims 1, 9, 10 and 12 of the ‘520 patent; claims 1, 7, 9-12 and 15 of the ‘261 patent; claims 1, 15, 32 and 33 of the ‘789 patent; and claims 3, 5, and 7 of the ‘709 patent. It also appears in claim 1 of the ‘678 patent as part of the term “floating point processor.”

⁶ In short, and as discussed in detail below, “software” programs are no more part of a “processor” than the music stored on a compact disc is part of the CD player you use to listen to it.

processor decodes or ‘interprets’ instructions and executes those decoded instructions”). With that understanding, PSI is willing to adopt the second part of IBM’s proposed construction (*i.e.*, everything after the word “that”), so that PSI is willing to agree to the following construction for processor: “One or more integrated circuits that interprets (*i.e.*, decodes) and executes instructions.” Thus, the Court need only address the single issue of whether a processor: (1) is composed of “one or more integrated circuits” or (2) should be construed as encompassing anything that is “a portion of a computer system.”

The Court should begin its analysis of this question by noting that neither party contends that “processor” is expressly defined in any of the patents. Instead, the parties agree that (1) since the time of the patents-in-suit, “processor” has had a well known and commonly understood meaning in the art, and (2) consistent with that fact, “processor” should be construed to have the same meaning across all of the patents. With respect to what that meaning is, the parties agree on two things: (1) a processor is a part of a computer system (as are every single one of the 78 other claim terms at issue in this lawsuit); and (2) a processor decodes and “executes instructions.”

IBM takes the position that the term processor “encompasses but is not limited to one or more integrated circuits.” IBM Br. at 16. **IBM, however, fails to provide even a single example, from either the intrinsic or the extrinsic evidence, of a processor that does not include one or more integrated circuits, or that comprises something other than an integrated circuit.** Furthermore, IBM feigns ignorance of the elementary computing term “integrated circuit” and states that “the phrase ‘integrated circuit’ is not found anywhere in the specifications of the ‘495, ‘789, ‘709 or ‘261 patents.” *Id.* at 13. But that statement, while literally true, is irrelevant: the phrase “portion of a computer system” is not found anywhere in *any* of the 10 patents-in-suit. By comparison, “integrated circuit” is used multiple times in the

'520 specification in a manner that supports PSI's construction,⁷ and "circuit" and "circuitry" are used multiple times in the specifications of the '495, '789 '678 and '709 patents in a manner supportive of PSI's construction.⁸

Specifically, the '520 patent, which claims an address translation technique for use in a processor during emulation, states the following in its specification:

When implemented as a PowerPCTM processor, each CPU 4 preferably comprises a single **integrated circuit** superscalar microprocessor, including various registers, buffers, execution units, and functional units that operate according to reduced instruction set computing (RISC) techniques.

'520 patent at 4:19-23 (emphasis added). While the specification does not define "integrated circuit," that term was well known in the art. So well known, in fact, that IBM repeatedly defined and used it in its own dictionary:

integrated circuit: A small piece of semiconductive material that contains interconnected miniaturized electronic circuits. Synonymous with microchip.

microchip: (1) Synonymous with integrated circuit. (2) A small piece of semiconductive material, usually silicon, that contains miniaturized electronic circuits. See also microprocessor.

microprocessor: (1) A processor whose elements have been miniaturized into one or a few integrated circuits. (2) A microchip containing integrated circuits that executes instructions.

processor: (1) In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (2) One or more integrated circuits that process coded instructions and perform a task.

IBM Dictionary of Computing (1994) at 347, 432-33, 533.⁹ In other words, integrated circuits are what processors are made of. See Declaration of Dr. Yale Patt, filed concurrently herewith

⁷ See, e.g., '520 patent at 4:19-23 ("When implemented as a PowerPCTM processor, each CPU 4 preferably comprises a single **integrated circuit** superscalar microprocessor..."); *PowerPC 604 RISC Microprocessor User's Manual* (Ex. 7) at ii (using "integrated circuit" as a synonym for "microprocessor"), incorporated by reference into the '520 specification at col. 1:57-59.

⁸ See, e.g., the 1994 edition of the *ESA/390 IBM Principles of Operation* ("390 POO"), incorporated into the specifications of the '495, '789, '106, '678 and '709 patents, at 5-70 and 11-2. A copy of all pages of the 390 POO cited in this brief is attached as Ex. 8.

(hereinafter “Patt Decl.”), at ¶ 6. Indeed, IBM’s own technology tutorial in this lawsuit concedes this point:

Processors include millions of tiny circuits. Today, and since the 1960s, these circuits are formed in semiconductor material such as silicon.¹⁰

While the specifications do not expressly define the term “processor,” their discussions of processors make clear that they all use the term “processor” in its ordinary, physical (*i.e.*, integrated circuit) sense. First, PSI notes that the patents use “processor” interchangeably with “CPU” and “microprocessor,” as IBM does in its brief. *See, e.g.*, ‘520 patent at 1:50-57; ‘495 patent at 7:21-26; ‘261 patent at 8:37-39; ‘789 patent at 7:10-30; ‘709 patent at 2:10-18; and IBM Br. at 13-16. With that in mind, PSI directs the Court to the 1994 edition of the *ESA/390 IBM Principles of Operation* (“390 POO”). This 600-page document is intrinsic evidence because it is incorporated by reference in the ‘789, ‘106, ‘678, ‘709 and ‘495 patents.¹¹ The 390 POO contains a section entitled “CPU,” which makes clear that a CPU (*i.e.*, processor) is hardware and does not include software programs:

The **physical implementation of the CPU** may differ among models, but the logical function remains the same. The result of executing an instruction is the same for each model, providing that the program complies with the compatibility rules.

The CPU, in executing instructions, can process binary integers and floating-point numbers of fixed length, decimal integers of variable length, and logical information of either fixed or variable length. Processing may be in parallel or in series; **the width of the processing elements, the multiplicity of the shifting paths**, and the degree of simultaneity in performing the different types of arithmetic differ from one CPU to another without affecting the logical results.

To perform its functions, the CPU may use a certain amount of internal storage. Although *this internal storage* may use the same *physical* storage medium as main storage, it is not considered part of main storage and *is not addressable by programs*. **The CPU provides registers** which are available to programs but do

⁹ A copy of all pages of the *IBM Dictionary of Computing* (1994) cited in this brief is attached as Ex. 9.

¹⁰ These sentences (verbatim) are part of the “Processors” portion of IBM’s June 1, 2008 Technology Tutorial CD.

¹¹ *See* ‘789 patent at 5:11-14, ‘709 patent at 1:15-17, ‘678 patent at 1:14-17, ‘106 patent at 1:15-17, and ‘495 patent at 2:2-4.

not have addressable representations in main storage. They include the current program-status word (PSW), the general registers, the floating-point registers, the control registers, the access registers, the prefix register, and the registers for the clock comparator and the CPU timer.

390 POO at 2-2 (emphasis added).

IBM cites the first sentence in the passage quoted above for the proposition that “it is the logical function of the processor (rather than its physical implementation) that is relevant.” IBM Br. at 15-16. But that argument misses the point: no one disputes the tautological proposition that processors following the same logic (e.g., “If A=B & B=C, then A=C”) will yield the same results when they execute the same instructions (e.g., A+B = 2C). The point is that while logic is abstract, CPUs (*i.e.*, processors) *must* have a “physical implementation” to execute instructions that follow that logic.¹² The fact that this physical implementation “may differ among models” is irrelevant—the point is that *all* models have one. And that physical implementation, of course, is the integrated circuitry that is found in all processors, which is why the 390 POO mentions “the *width* of the processing elements” (*i.e.*, the width of the circuits) and “the multiplicity of the shifting paths” (*i.e.*, the shifting paths in the circuitry). 390 POO at 2-2.

The intrinsic evidence quoted above also makes clear that CPUs (*i.e.*, processors) and “programs” are distinct concepts: among other things, they each have different capabilities with respect to “internal storage.” *Id.* The intrinsic evidence also asserts in unqualified language that a “CPU provides registers which are available to programs but do not have addressable representations in main storage, includ[ing] the current program-status word (PSW) [and] the general registers.” *Id.* “Register” and “program status word” are two disputed claim terms discussed below. For present purposes, however, it is worth noting that the intrinsic evidence establishes that (1) a CPU “provides” registers (which IBM acknowledges are “internal storage

¹² Indeed, as used by computer scientists, the term “logic” actually *means* the physical circuitry used in processors. See *IBM Dictionary of Computing* (1994) at 396 (defining “logic” as “The systematized interconnection of digital switching functions, circuits or devices.”).

having a specified storage capacity,” *see* IBM Br. at 23) and (2) the PSW is physically located in the CPU—two facts further supporting PSI’s hardware construction of “processor.”

IBM’s principal argument concerning “processor” is the straw man contention that PSI’s proposed construction “excludes the possibility of using microcode” and therefore reads out the preferred embodiments of the ‘495, ‘789 and ‘261 patents. IBM Br. at 14-15. This argument lacks merit because it is based on two false premises: (1) “microcode” is the same thing as “software” and (2) “microcode” cannot be a component of an integrated circuit. Specifically, IBM’s brief equates “microcode” with “software”—*see* IBM Br. at 14, stating “microcode (*i.e.*, software)” —and assumes that “microcode” must be distinct from an “integrated circuit.” *See id.* (“PSI’s construction...excludes the possibility of using microcode”). IBM provides no discussion or authority to support those two premises—nor could it, since both are false.

The intrinsic evidence, in passages that IBM selectively ignores, expressly differentiates between “microcode” and “software”:

As already noted, the RP instruction described above, like other aspects of the CPU architecture, may be implemented by [1] **hardware**, by [2] **microcode**, or by [3] any suitable **combination of the two**, *while* the signal catcher 118 utilizing the RP instruction is preferable implemented as [4] **software**. (Both microcode and software constitute programming, **the principal difference being that microcode implements an architectural interface while software interacts with it.**)

‘495 patent at 11:9-16 (emphasis added). This passage makes two key points. First, “microcode” and “software” are not identical, as IBM assumes. Second, microcode *implements* an architectural interface (*i.e.*, which is what a processor does), rather than *interacts* with one (*i.e.*, which is what a software program does). In short, microcode is necessarily *part of* the processor.

Further support for PSI's construction is provided by the fact that the intrinsic evidence explains at length the immutable nature of microcode and expressly states that microcode is a *"manufactured component"* of the processor:

As a preliminary to describing the invention itself, the insights that led to it will be discussed. Problem state programs generally have the least authority among the agencies of the computing system. Only a subset of the architected facilities are available to it. The defined architecture of the system provides the operating system with a set of architected facilities which it may use, and in some cases allocate to problem state programs. Some of those that are withheld from direct use are made available by means of system services provided by the operating system, particularly where physical resources are shared by different programs, e.g., real main storage, external storage space, networking facilities, etc. In like manner, at a lower level of control, the microcode and hardware agencies of the system have defined facilities for their own use which are not accessible, or even seen at the system architecture level. Examples are a *section of storage not accessible to programs* in either problem or supervisor state, which storage is required to perform the invocable functions of the architecture, *registers reserved for internal use*, adders and other *logical units* needed to do addressing and arithmetic. The microcode in particular must perform its assigned programming tasks without polluting the architected facilities of the system. It operates at a third, separate and isolated, level of control in the system, with capabilities beyond the problem state and the supervisor state. The microcode and hardware of the system are designed and the design verified and tested for correctness before the system is manufactured. Special concern is paid that the hardware and microcode can not be compromised as far as system integrity is concerned by actions performed by programs in either supervisor or problem state. Otherwise, the authority structure of the system architecture can not be guaranteed.

The major point here is that the microcode does not change dynamically, it can be considered to be a manufactured component of the system, and is a fully trusted element in the integrity structure of the system. The microcode has full access to programmable storage, and to the Program Status Word (PSW) of the system as part of its necessary capabilities. The microcode can perform complex operations of many steps in order to provide the functions of what is a single instruction at the architecture level above it. Similarly, below the microcode level of control, the hardware elements of the system must have complete access to all system facilities, even those not readily available to the microcode.

'495 patent at 4:23-67 (emphasis added). Again, PSI *agrees* that microcode can be (and often is) part of the processor. But the intrinsic evidence makes clear that IBM's position—microcode is a type of software, so that means *all* software can be part of the processor—lacks merit. As

something that has “defined facilities for [its] own use which are not accessible or even seen at the system architecture level”—including “a section of storage not accessible to programs,” “registers reserved for internal use,” and “logical units”—microcode is *different* from software programs. It is, instead, *part of* the one or more integrated circuits that constitute the processor—or, as the intrinsic evidence puts it, microcode is “a *manufactured component* of the system.” ‘495 patent at 4:57-58 (emphasis added).

This use of “microcode” in the intrinsic evidence reflects the common understanding of “microcode” to those of ordinary skill in the art. Professor Yale Patt, for example, explains “microcode” as follows in his declaration:

Microcode is not “software”—it is part of the hardware of the computer. Microcode has some similarities to software (namely the fact that it can be written out in code), but it is a *manufactured component* of the hardware, not a software program that can be loaded and/or modified (which is what makes software “soft”). Since before the 1990s and all the way through today, microcode has been stored in unchangeable circuits *within* the larger integrated circuits that make up the processor. Indeed, if you receive (for example in the mail from Intel) a microcoded x86 processor and you were to somehow *remove* the microcode (for example, by cutting out the read-only circuits that store it), you would no longer have an x86 processor. The point is this: microcode is seen by people in the art as being a manufactured component of the hardware because, unlike software, it cannot be changed and instead comes burned into the chip like any other logic element.

Patt Decl. ¶ 7 (emphasis in original). As the Court may recall, in April 2008 IBM invited Dr. Patt to serve as the Court’s technical advisor in this action (Ex. 12).

Notably, IBM is unable to cite anything in the intrinsic evidence that contradicts PSI’s construction of “processor,” or that contradicts the above discussion of microcode. Indeed, the only passages from the ‘261 patent that IBM is able to cite both *support* PSI’s position:

This preprocessing can be provided by means of a hardware preprocessor, or be implemented in microcode, or simply be performed by a program executing on the target processor. In the embodiment shown herein, it is a program....The preferred embodiment described in detail herein uses a *microcoded implementation in the target processor*. The microcode is software *which is protected from being changed* by user software executing in the target processor.

'261 patent at 4:14-18, 8:61-65 (emphasis added). Thus, microcode is distinct from "user software," and "user software" is not *part of* the processor (as IBM would have it) but, instead, is something that the processor *executes*.

Finally, IBM's own contemporaneous dictionary also supports PSI's position:

microcode: (1) One or more microinstructions. (2) A code, representing the instructions of an instruction set, that is implemented in a part of storage that is not program-addressable.

Note: The term microcode represents microinstructions used in a product as an alternative to hardwired circuitry to implement functions of a processor or other system component.

microinstruction: An instruction for operations at a level lower than machine instructions.

IBM Dictionary of Computing (1994) at 432. In short, IBM's claim construction argument for "processor" misses the point because it is built on an obvious non sequitur: (1) microcode can be part of a processor, (2) microcode can be categorized as a type of software, therefore (3) *all* software can be part of a processor. That is no different than saying (1) birds can fly, (2) birds can be categorized as a type of animal, therefore (3) all animals (*e.g.*, pigs) can fly. The Court should reject both IBM's false logic and its unbounded construction.

IBM's second argument for allowing software to become part of a processor is its contention that in the preferred embodiment of the '520 patent "semantic routines," which IBM also equates with "software," are part of the processor, and that PSI's construction therefore must be rejected because it excludes "semantic routines." IBM Br. at 14-15. This argument fails on its face because the single portion of the '520 specification that IBM quotes expressly states that the processor is not comprised of semantic routines but, instead, *executes* them:

When implemented as a PowerPCTM processor, each CPU 4 preferably *comprises* a single *integrated circuit* superscalar microprocessor....Each CPU 4 is further adapted in accordance with the present invention to execute guest instructions (*e.g.*, CISC instructions or some other instruction set that is not native to CPU 4)

by emulation. As described further hereinbelow, guest instructions 20 are each emulated by fetching and **executing** one or more **semantic routines** 19, which each contain two or more native instructions.

Id. at 14-15, quoting ‘520 patent at 4:19-45 (emphasis added). Under IBM’s construction a processor literally expands and shrinks over time: at one point, it may include only “CPU 4”; but later, depending on whatever software program the user happens to run on CPU 4, it might also include portions of that software, such as “semantic routines 19.” But the intrinsic evidence rejects this bizarre shape-shifting view of a processor, expressly stating that a “processor” is “**comprised of**” an “integrated circuit” and that it “**executes**” semantic routines. ‘520 patent at 4:19-45; *see also id.* at 16:11-18:34 (reflecting the fact that **all 16 claims** of the ‘520 patent refer to a “processor” that “**execut[es]** a semantic routine”).

Finally, the extrinsic evidence concerning “processor” overwhelmingly supports PSI’s “integrated circuit” construction. As indicated above, PSI’s proposed construction is taken *verbatim* from the contemporaneous *IBM Dictionary of Computing*:

processor: (1) In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit. (2) ***One or more integrated circuits that process coded instructions and perform a task.***

IBM Dictionary of Computing (1994) at 533 (emphasis added). While IBM criticizes PSI for relying on “the narrower dictionary definition [2],” the *IBM Dictionary of Computing* expressly states that definition [1], which IBM relies on, is “taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1,” and it notes that “final agreement has not yet been reached” on definition [1]. *Id.* at vii-viii. Definition [2] (*i.e.*, PSI’s proposed construction), therefore, is the **only** official contemporaneous definition actually authored by IBM.

Apart from its dictionary, IBM’s own patent license with [REDACTED], expressly defines “processor” as follows:

(Ex. 10 at 4). Like IBM's contemporaneous dictionary, IBM's patent license supports PSI's construction.

PSI also respectfully directs the Court to the declaration of its expert Dr. Yale Patt, which emphatically rejects IBM's contention that a "processor" can include software programs and which supports PSI's proposed construction.

The Court should reject IBM's disingenuous "portion of a computer system" construction and should adopt PSI's proposal.

B. "Instruction"¹³

IBM's Proposed Construction	PSI's Proposed Construction
A language construct that specifies an operation and identifies its operands, if any.	A string of digits that specifies an operation and identifies its operands, if any, and can be directly executed by the processor to which it is directed.

There are two issues that need to be decided with respect to this claim term: whether an "instruction" (1) is a "language construct" or a "string of digits" and (2) is something that can be directly executed by the processor to which it is directed. IBM expressly acknowledges in its brief that in making its proposed construction it is "relying mainly on extrinsic evidence"—and, indeed, **IBM does not cite a single instance of support from the intrinsic evidence for its construction.** See IBM Br. at 17-19. IBM ignores the extensive use of "instruction" in the intrinsic evidence because that evidence overwhelmingly supports PSI's construction and flatly rejects IBM's awkward "language construct" proposal.

1. "Language construct" versus "string of digits"

¹³ The term "instruction" appears in claims 1, 3, 6-12 of the '709 patent; claims 1-4 and 9-12 of the '520 patent; claims 1-3, 5-11, 13-16, 18-21, and 23 of the '495 patent; claims 4, 8, 13, 15, 23 and 31 of the '789 patent; claims 1, 3, 5, 7, 9, and 11 of the '678 patent; and claims 1-14 and 16-47 of the '261 patent.

IBM objects to “string of digits” as too narrow and argues that “instruction” should include “language constructs”—the example IBM gives in its brief is “ADD A,B.” IBM Br. at 17. While “ADD A,B” may be easier for humans to work with, it is gibberish to a computer’s processor. Patt. Decl. ¶ 14. Even IBM’s expert acknowledges that such language constructs “*require translation into the corresponding machine instruction(s) at the machine language level before it (they) can be executed by a processor.*” Smotherman ¶ 10 (emphasis added). And “machine instructions,” IBM acknowledges, are made up of a “string of digits.” IBM Br. at 54.

The claims of the patents-in-suit establish that an “instruction” is something that can be executed by a processor, as the following representative examples demonstrate:

- ‘709 patent: “...a *processor* that *executes* said *instruction* to provide a result of a given accuracy” (claim 9); “...said default rounding mode is operative during *execution* of said *instruction* by said *processor*” (claim 10); “...a *processor* that *executes* a floating point *instruction*” (claim 1).
- ‘495 patent: “In an information handling system in which *execution* of a program of *instructions* by a *processor*...” (claim 1).
- ‘789 patent: “...wherein the sequence value indicates a correct sequence of execution of an instruction, regardless of which *processor* of the one or more processors *executes* the *instruction*” (claim 15).
- ‘261 patent: “An emulation method for executing an incompatible computer program on a target processor, the incompatible program containing computer *instructions* natively *executable* on a different *processor*...target *instructions executable* on the target *processor*.” (claim 1).
- ‘520 patent: “A method of operating a *processor*...comprising...storing in memory a semantic routine of native *instructions*...and *executing* a semantic routine.” (claim 1).

Construing “instruction” to include “language constructs” like “ADD A,B” would render the patent claims in this lawsuit nonsensical, since it is undisputed that a “language construct” cannot be executed by a processor.

Like the claims, the specifications also demonstrate that an “instruction” is something that can be executed by a processor, and is therefore necessarily composed of a string of digits or “bits”:

- ‘709 patent: “FIG. 1 illustrates a conventional shared memory computer system including a plurality of central processing units (CPUs) 102-108 all having access to a common main storage 110. FIG. 2 schematically depicts functional components included in a CPU from FIG. 1. Instruction unit 200 fetches *instructions* from common main storage 110 according to an instruction address located in the program status word (PSW) register 202, and appropriately *effects execution of these instructions*.” 2:10-18.
- ‘495 patent: “One of the key mechanisms in an S/390 system is the program status word (PSW), which directs the *processor* in the execution of a program. It indicates the next *instruction* to be *executed*...” 2:5-8.
- ‘789 patent: “Two *executions* of a STORE CLOCK *instruction* (used by a program to obtain the date and/or time), possibly on different central processing units (CPUs), are to store different values....This is true whether the two *instructions* are *executed* on the same or different *CPUs*.” 1:44-55.
- ‘678 patent: “In FIG. 5 is illustrated *a typical instruction format* as is used, e.g., in an IBM System 390 computing system....In BFP mode, the rightmost 12 *bits* of the address, *bits* 20-31, are used to specify 12 combinations of operand class and sign.” 3:19-39 (emphasis added).

In addition, the ‘678 patent uses the terms “machine instruction” and “instruction” interchangeably, referring in the claims to “a *machine instruction* to determine whether the data class of a floating point number is an identified data class,” and in the specification to the same instruction as “the Test FP Data Class *instruction*.” ‘678 patent at 1:61, 4:63-64 (emphasis added). The ‘261 patent does the same thing. *See, e.g.*, ‘261 patent at 2:40-41, 13:47-54 (referring to interchangeably to “instructions” and “machine instructions” of a particular computer system). As noted above, IBM acknowledges that a “machine instruction” is composed of a “string of digits.” IBM Br. at 54.

Finally, the 390 POO (intrinsic evidence because it is incorporated by reference into the specifications of the ‘789, ‘709, ‘678, ‘106 and ‘495 patents) contains a section entitled “Instructions” which indicates in unqualified language that “instructions” are comprised of bits:

The first byte or...the first two bytes of an instruction contain the op code. For some instructions in the S format, all or a portion of the second byte is ignored. The first two bits of the first or only byte of the op code specify the length and format of the instruction.

390 POO at 5-4. In sum, like the claims, the specifications all make clear that an "instruction" is something that can be executed by a processor and that is therefore composed of a string of digits.

The intrinsic evidence alone, as recounted above, establishes that IBM's "language construct" proposal is nonsensical and that PSI's "string of digits" construction should be adopted. To the extent the Court even finds it necessary to look to the extrinsic evidence, IBM's 1999 Claim Construction Brief in *Data General Corp. v. IBM*, 94-cv-12213 (D. Mass.), is on point and squarely states the following:

IBM contends that the word "instructions" means *executable instructions* into which programs written in high-level user languages are *compiled for execution by the processor*.

IBM's *Data General* Brief (Ex. 4) at 42 (emphasis added). While IBM will argue that its *Data General* brief applies only to the patents in that action, the generally applicable technological discussion that brief provided in support of the above construction belies that argument:

The function of a digital data processing system (or "computer system") is to run programs by executing instructions that the system's processor can understand. The set of all instructions executable by a processor is referred to as its "machine instruction set" or "machine language." Machine instructions consist of binary code (*i.e.*, strings of 1's and 0's), which is all that a processor can actually understand or execute.

In the earliest computers, programmers wrote their programs directly in machine language. Because human beings do not think intuitively in binary form, writing programs in the form of 1's and 0's was extremely time consuming and difficult. To overcome this difficulty, programmers developed various high level user programming languages, such as FORTRAN and COBOL. These languages were "high level" in the sense that they closely resembled human language rather than the stream of binary digits to which a computer processor was able to respond.

While high level languages were easier for humans to use, they could not be understood or executed by a computer. Thus, in order to run a high level language program on a computer, the high level language statements first had to

be translated (or “compiled”) into machine language instructions, which a processor could understand and execute. This translation was generally performed by a computer program known as a “compiler.”

Id. at 20, 42 (emphasis added). Indeed, even in the present lawsuit, IBM’s proposed construction for “processor” recognizes that a processor “executes instructions”—IBM saw no need to instead say “executes *machine* instructions.” Thus, the extrinsic evidence further bolsters PSI’s construction.

2. “...directly executed by the processor to which it is directed”

The other dispute with respect to “instruction” is whether “instructions” should be defined relative to the processors that handle them (PSI’s position) or in an absolute manner that ignores that relationship (IBM’s position). To put it another way, PSI’s proposal treats the term instruction *from the perspective of a processor (i.e., by defining an instruction in terms of what a processor recognizes and treats as an instruction)* while IBM attempts to define the term *in the abstract*. But PSI’s position is clearly right, and it is overwhelmingly supported by the intrinsic evidence. Indeed, that is why IBM’s argument on this term does not contain so much as *one* intrinsic citation. *See IBM Br.* at 17-19.

First, it is undisputed that a processor can only understand and execute the instructions it is *designed* to recognize. In IBM’s own words, “Intel based computers...are *incapable* of understanding and executing IBM instructions.” *Id.* at 7. Or, to put it another way, the only things a processor recognizes *as an instruction* are the things that are *defined* as instructions in the Instruction Set Architecture to which that processor is designed.¹⁴ *See* Patt Decl. ¶ 12. From the processor’s perspective, *anything* other than the instructions defined in its ISA is *data* to be acted upon and not an “instruction” that it can understand and execute. *Id.* Indeed, the whole point of an emulator, such as those claimed in the ‘520 and ‘261 patents, is to translate

¹⁴ IBM’s technology tutorial makes this same point: “All of the instructions that a processor is designed to understand and execute make up the ‘instruction set’ for that processor.” (Ex. 11).

incompatible or “*guest* instructions” from what they are (*i.e.*, data) into *instructions* that a particular processor can recognize and execute. The ‘520 inventor expressly acknowledged this point:

[REDACTED]

Mallick Depo. (Ex. 3) at 105:15-24 (emphasis added, form objection omitted).

This does *not* mean, however, that “guest instructions” or “incompatible instructions,” as those terms are used in the ‘520 and ‘261 patents, are not instructions at all. Indeed they are. But they are only instructions *with respect to* those processors that can directly execute them—to all other processors, as the ‘520 inventor recognized, they are *data*. PSI’s construction recognizes this distinction between instructions and data and treats the “guest”/“incompatible” instructions appropriately: (1) as “instructions” relative to the “guest”/“incompatible” processors that can execute them and (2) as “data” relative to the “host”/“target” processors that cannot. And this is how the ‘520 and ‘261 patents treat these terms, as both patents claim methods of translating “guest instructions” and “incompatible instructions” into instructions that the claimed processors can actually execute. *See, e.g.*, ‘520 patent at 16:11-31 & 17:5-23; ‘261 patent at 19:22-20:11 & 20:62-21:37.

Finally, the idea that instructions are strings of digits that can be directly executed by the processor is reflected in the fact that, as used in the patents, the term “instruction” is synonymous with “machine instructions.” In addition to the uses of “machine instruction” in the ‘261 and ‘678 patents discussed above, note the following:

- The ‘520 patent states that the relevant “instructions...can be characterized as having a fixed instruction length (e.g. 32 bits)...and being executed without microcoding, often in one machine cycle.” 4:32-38. The fact that the ‘520 talks

about instructions as having a *bit length* and being executed during one *machine cycle* indicates that the instructions to which it refers are machine instructions.

- The '709 patent states that "[i]nstruction unit 200 fetches *instructions* from common main storage 110 according to an instruction address located in the program status word (PSW) register 202, and appropriately *effects execution of these instructions*." 2:10-18. Instructions that are "fetched" and "executed" can only be machine instructions.
- The '789 describes one aspect of the invention as "embodying at least one program of *instructions executable by the machine* to perform a method of generating sequence values." 3:62-63.
- '495 patent: "One of the key mechanisms in an S/390 system is the program status word (PSW), which directs the *processor* in the execution of a program. It indicates the next *instruction* to be *executed*..." 2:5-8. The instruction designated by the PSW for "execution" can only be a machine instruction.

The parties agree that "machine instructions" can be "directly executed by a processor." IBM Br. at 54.

As the foregoing discussion makes clear, when the patents talk about "instructions" they are talking about things that can be *directly executed* by a processor. This is because instructions are only instructions if the processor can see and understand them as such. In other words, the patents themselves establish that the concept of an "instruction" is one that is *relative to* a processor, and not just some abstract concept. If it were otherwise, then every English language command, such as "print this brief," would qualify as an instruction. Indeed, under IBM's abstract construction "print this brief" *does* qualify as an "instruction," since it is a "language construct" that both "specifies an operation" (print) and "identifies an operand" (the brief). Yet it would be *absurd* to say that the patents intended to include this kind of statement within the concept of "instruction." And they didn't—they were instead talking about *what a processor recognizes as an instruction*. That means they were talking about executable, machine instructions—just as IBM claimed in its *Data General* brief.

For these reasons, the Court should adopt PSI's proposal for "instruction."

C. “Register”¹⁵

IBM’s Proposed Construction	PSI’s Proposed Construction
A part of internal storage having a specified storage capacity and usually intended for a specific purpose.	A hardware storage element in the processor that can be accessed faster than memory.

The principal issue the Court must decide with respect to this claim term is whether a “register” refers to: (1) a specific type of storage element (PSI’s position); or (2) every storage device in a computer (IBM’s position). IBM admits that it is “relying predominantly on extrinsic evidence to construe the term” and **does not cite a single piece of intrinsic evidence to support its unbounded construction**, which literally captures every single storage device inside a computer. IBM Br. at 23-24. IBM runs from the intrinsic evidence because the intrinsic evidence overwhelmingly rejects IBM’s unbounded construction and supports PSI’s proposal.

1. “Hardware” and “in the processor”

PSI’s proposed construction says both *what a register is* (a hardware storage element in the processor) and what distinguishes it from the other storage elements in a computer (the fact that it can be accessed much faster than other forms of storage). With respect to what a register is, there are two basic points: (1) it is hardware and (2) it is *in* the processor.

The first of these points should be non-controversial, since IBM acknowledges that a register is “a part of internal storage having a specified storage capacity.” You could no more build a register out of 0s and 1s (*i.e.*, software) than you could build a bookshelf out of words. In short, as a storage device a register must be *hardware*. Indeed, the ‘520 inventor was unequivocal about this fact during his deposition¹⁶:

██

¹⁵ The term “register” appears in claims 2-4, 7, 10-11, 15-17 and 19 of the ‘495 patent and claims 13 and 31 of the ‘789 patent, as well as in the specifications of all of the patents-in-suit.

¹⁶ Although “register” does not appear in the ‘520 claims, the parties have agreed that “register” should be given a common meaning for the patents-in-suit, and indeed “register” is used in the same way in all of the patents. Thus, the ‘520 patent and its use of register is persuasive evidence for the construction of this term.

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

Mallick Depo. (Ex. 3) at 59:15-60:3 (form objections omitted).

As to whether a register is *in* the processor, the intrinsic evidence demonstrates overwhelmingly that it is. For example, the 390 POO, which is part of the specifications of the both the '495 and the '789 patents, contains a chapter on the "Organization" of a computer system. The table of contents of that chapter makes clear that, contrary to IBM's construction, a "register" is distinct from general "storage":

Chapter 2. Organization

Main Storage.....	2-2
Expanded Storage.....	2-2
CPU.....	2-2
PSW.....	2-3
General Registers.....	2-3
Floating Point Registers.....	2-3
Control Registers.....	2-4
Access Registers.....	2-4
Vector Facility.....	2-6
External Time Reference.....	2-6
I/O.....	2-6

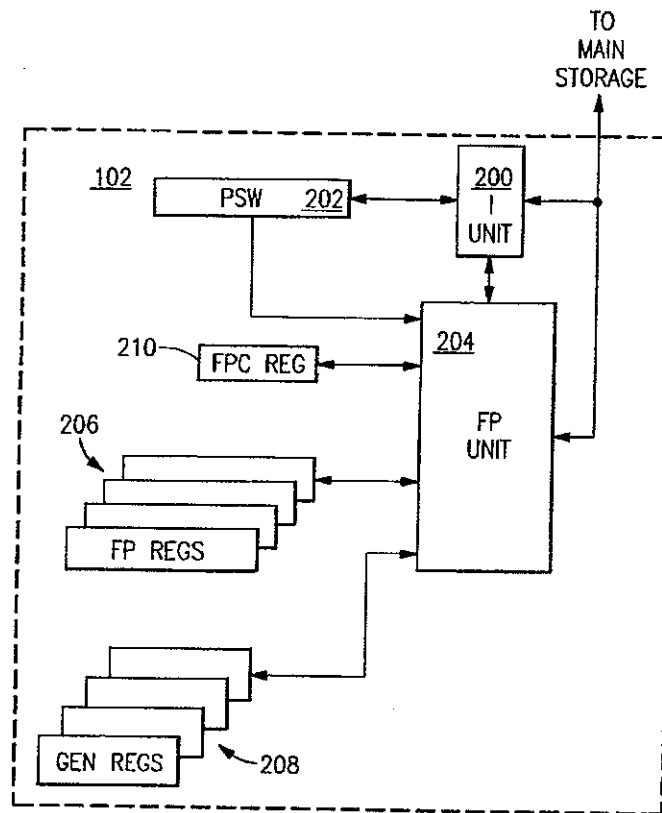
390 POO at 2-1. The 390 POO expressly states that “[t]he CPU provides registers,” and refers to registers as “internal storage” in the CPU. *Id.* at 2-2. The ‘495 specification also refers to “the CPU registers” (col. 1:10), while the ‘789 specification expressly refers to a register as being “in the CPU.” ‘789 patent at 9:50. In addition, the ‘520 expressly states that “When implemented as a PowerPCTM processor, each CPU 4 preferably comprises a single integrated circuit superscalar microprocessor, *including various registers...*” ‘520 at 4:19-21. And, in contrast to IBM’s unbounded construction, the ‘789 claims distinguish between a “register” and a general “storage area”:

5. The method of claim 4, wherein said providing comprises retrieving, by said instruction, said timing information from a physical clock, and wherein said including comprises retrieving, by said instruction, said selected information from a storage area.

6. The method of claim 5, *wherein said storage area is a programmable register* set by a set register instruction.

‘789 patent at 40-46 (emphasis added). IBM’s construction provides no meaningful distinction between a “register” and any other storage area—and in failing to do so it is contrary to the intrinsic evidence.

The specifications also *depict* registers as being inside the processor. For example, in the ‘709:



See '709 patent at Figure 2 & 1:58-60 ("Fig. 2 schematically depicts functional components included in a CPU"); *see also* '678 patent at Figure 2 & 2:15-16 ("Fig. 2 [the same image] illustrates functional components included in a CPU"); '106 patent at Figure 1 (showing "A Register," "B Register," and "C Register" *within* the floating point unit *inside* a processor); '520 patent at Figure 4 & 7:47-48 ("As depicted in FIG. 4, guest instruction queue 100 has an associated emulation instruction pointer *register*"), and Figure 2 & 3:46-47 ("Fig. 2 depicts a more detailed block diagram of the processor" and contains the emulation assist unit shown in Figure 4).

Similarly, when asked where the registers mentioned in the '520 patent were located, the '520 inventor was unequivocal:

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]

Mallick Depo. (Ex. 3) at 90:12-22 (emphasis added).

Finally, in contrast to PSI's construction IBM's construction has three parts but it tells you *nothing* about what a register actually *is*: "A part of internal storage [this could be anything inside the computer, including the hard drive] having a specified storage capacity [all storage is finite, therefore all storage devices inside a computer have a "specified storage capacity"—this could still be the hard drive] and usually intended for a specific purpose [what specific purpose? and, in any case, the qualifier "usually" renders this limitation meaningless—and this could *still* be the hard drive]." While IBM would like an unbounded construction so that it can roam around inside PSI's computer systems and call literally anything a "register," the evidence demonstrates that registers are distinguishable from other internal storage devices, including in particular by the fact that they are *in* the processor.

2. "can be accessed faster than memory"

Consistent with its ordinary meaning, the patents use "memory" to refer to main memory, which is located *outside* of the processor. *See, e.g.*, '789 patent at Figs. 12 & 13; '678 patent at Fig. 1 & 2:13-14; '709 patent at Fig. 1 & 1:55-57. One of the reasons registers must be located *inside* a processor is that they are used for critical tasks like telling the processor what instruction to execute next. *See* '495 patent at 2:5-8 (stating that the "PSW...directs the processor in the execution of a program" and "indicates the next instruction to be executed") and 390 POO at 2-3 (stating that the PSW is a "register"). For this reason, at the time of the patents-in-suit (*i.e.*, the

1990s-2005), one of ordinary skill in the art would have considered it *absurd* to put a register in memory. Patt Decl. ¶ 21. By 1997 (the year the '495 patent application was filed), registers (*i.e.*, the ones described in the intrinsic evidence as being *in* the processor) could be accessed 100 times faster than main memory—putting a register in main memory would literally make the computer trying to access the subject data run 100 times slower. *Id.* For this reason, the intrinsic evidence recognizes that registers are “localized *high-speed* memory.” ‘106 patent at 3:55-56 (emphasis added). And for this reason one of ordinary skill in the art in the 1990s simply would not have understood “register” to refer to anything other than a high-speed hardware storage element in the processor. *See* Patt Decl. ¶ 22. The following extrinsic cites are instructive:

- IBM’s official terminology, which defines a “register” as “[a]n internal computer component capable of storing a specified amount of data and accepting or transferring this data *rapidly*.” *See* <http://www-306.ibm.com/software/globalization/terminology/qr.jsp> (Ex. 11).
- The *Encyclopedia of Computer Science* (at 1159) defines a “register” as “a specialized storage element of a CPU that consists of digital storage elements that respond *faster than those typically used to implement main memory storage locations*.” (Ex. 12).
- The Free On-Line Dictionary of Computing defines a “register” as “One of a small number of high-speed memory locations in a computer’s CPU.” <http://foldoc.org/index.cgi?query=register> (Ex. 13).

IBM acknowledges that “a register is typically a hardware component in a processor” and that registers are “accessed faster in many particular implementations.” IBM Br. at 24. But, without citing any supporting examples in the intrinsic evidence, IBM summarily argues that it is entitled to its “broader definition of the term.” *Id.* The single extrinsic example of a register IBM could dig up to oppose PSI’s construction is the old IBM System 360 Models 30 and 40, *which date from 1964*. *See* Smotherman Decl. ¶ 15. But the state of the art of computing in 1964 was not even remotely close to the state of the art in the mid-to-late 1990s. *See* Patt Decl. ¶ 21. The fact that the only counterexample IBM was able to dig up dates to 1964 *supports* rather than

detracts from PSI's proposed construction.

Finally, IBM makes the summary and inexplicable assertion that "a register could be implemented with a combination of hardware and software." IBM Br. at 24. The first and most glaring problem with this assertion is that it is inconsistent with IBM's own proposed construction: "storage," particularly "storage having a specified storage capacity," *cannot* be software (*i.e.*, zeros and ones); it *must* be hardware. The Court should reject this throw-away argument for that reason alone.

Second, the sole piece of evidence IBM relies on for its "hardware and software" argument is the same *1964* IBM System 360 Models 30 & 40. Specifically, according to Dr. Smotherman's report certain registers in those 1964 machines "were implemented in microcode (software) and not implemented in the hardware integrated circuits." Smotherman ¶15. But this assertion is both wrong and deliberately confusing. As noted above, you *can't* build registers out of zeros and ones—any more than you can build a bookshelf out of words. On the other hand, you *can* implement microcode in physical circuits that *output* the zeros and ones used to control the processor.¹⁷ What Smotherman's assertion does is try to blur the line between hardware and software by using the fact that microcode is implemented in hardware and can *sometimes* be considered software. IBM then takes that blurring and twists it to make assertions, like the one that "storage" can be made of "software," that are totally at odds with the common understanding of computing terms. Because IBM's awkward assertions are also totally at odds with the way "register" is used in the patents, IBM does not and cannot cite a single piece of intrinsic evidence to support them.

In sum, IBM's proposed construction completely ignores the intrinsic evidence and is based entirely on a single piece of extrinsic evidence that dates to 1964. PSI's construction,

¹⁷ Indeed, as noted *supra* in Section III.A, it is *precisely* because this is how microcode is implemented that it is *part of* the hardware that makes up the processor. *See also* Patt Decl. at ¶ 7.

which relies principally on the intrinsic evidence and which reflects the state of the art in the 1990s, should be adopted.

D. “Program Status Word”¹⁸

IBM’s Proposed Construction	PSI’s Proposed Construction
A defined set of data that indicates the next instruction to be executed and includes the program condition code and program authority, where that data directs the processor in the execution of a program.	The contents of the register that indicates the next instruction to be executed, includes the program condition code and program authority, and directs the processor in the execution of a program.

The parties agree on what the program status word (“PSW”) *does*—it “directs the processor in the execution of a program,” and, in particular, “indicates the next instruction to be executed.” The sole issue that the Court needs to decide is what the PSW *is*: (1) a “defined set of data” (IBM’s position) or (2) the contents of a specific register (PSI’s position). The intrinsic evidence demonstrates that processors have a “PSW register” (‘678 patent at 2:61) and that, by definition, it is the contents of this register that constitute the PSW. 390 POO at 2-7.

As discussed in the previous section, registers are high-speed storage elements located in the processor and used for critical tasks. The PSW “directs the processor in the execution of a program” and, in particular, “indicates the next instruction to be executed.” ‘495 patent at 2:6-8. The intrinsic evidence unambiguously establishes that the PSW is a register. The 390 POO, intrinsic evidence because it is incorporated by reference into the specifications of the ‘678 and ‘495 patents (as well as those of the ‘789, ‘709, and ‘106 patents), states in unqualified language that the PSW is a “register” in the CPU:

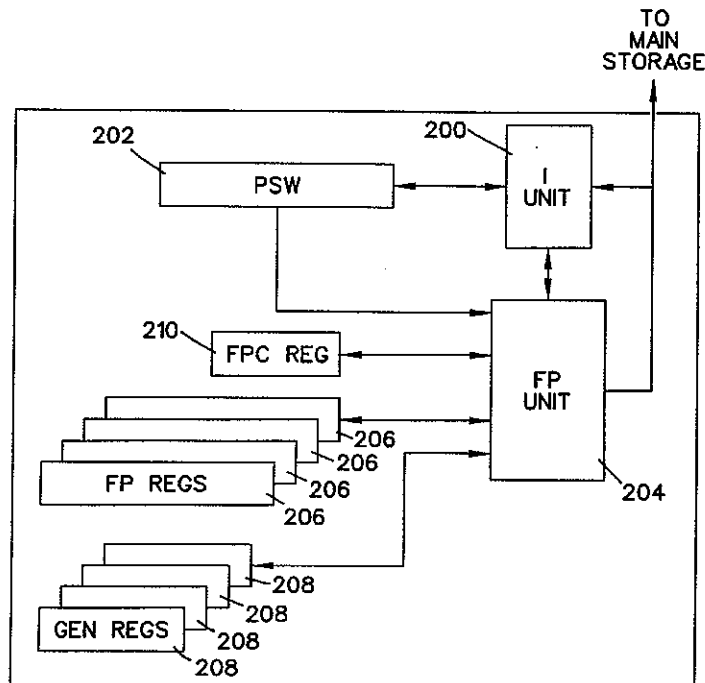
The CPU provides registers which are available to programs but do not have addressable representations in main storage. **They include the current program-status word (PSW)**, the general registers, the floating-point registers, the control registers, the access registers, the prefix register, and the registers for the clock comparator and the CPU timer.

¹⁸ “Program status word” is used in claims 1 & 7 of the ‘678 patent and claims 1, 6, 7, 12, 14, 19 & 22 of the ‘495 patent.

390 POO at 2-3. The '678 patent refers expressly to the "PSW registers":

Instruction unit 200 fetches instructions from common main storage 110 according to an instruction address located in the program status word (PSW) register...FIG. 3 illustrates the format of a 64 bit PSW as stored in PSW register 202.

'678 patent at 2:40-41, 60-61 (emphasis added). And, in Figure 2, the '678 patent even depicts the PSW as a structure parallel to other registers *in* the CPU:



The '678 patent states that this figure "illustrates functional components included *in* a CPU." 2:15-16 (emphasis added). The '709 patent contains a substantively identical illustration and description of a CPU with a PSW register inside it. '709 patent at Fig. 2 & 1:58-60.

Finally, consistent with the 390 POO (which is part of its specification) and with the '678 patent, the '495 patent also expressly refers to the PSW as being a "register":

In addition to registers 100 and 112 and PSW 114, CPU 102 has *other registers* (such as control registers and floating-point registers) that are not relevant to the present invention and are hence not shown.

'495 patent at 7:52-56 (emphasis added).

IBM acknowledges that “Figures 2 and 3 of the ‘678 specification each illustrate a program status word, and the ‘678 specification indicates that for those illustrations the program status word is stored in a register.” IBM Br. at 21. IBM argues, however, that claims 1, 7, 14 and 19 of the ‘495 patent indicate that the PSW can be relegated to the 100-times-slower main memory because those claims state that the PSW is located in a “storage location” or a “save area.” IBM Br. at 19. But that argument ignores both the ‘495 specification and the express terms of those claims. Claims 1, 7, 14 and 19 of the ‘495 patent each state that the “*saved* program status word,” and not *the* program status word *actually directing the processor*, is stored in the main memory “storage location” or “save area.” ‘495 patent at 11:34-35, 12:14, 13:17-18, 14:3-4 (emphasis added).¹⁹ IBM simply ignores this distinction. IBM Br. at 19-23.

As noted above, the PSW “directs the processor in the execution of a program” and, in particular, “indicates the next instruction to be executed.” ‘495 patent at 2:6-8. The ‘495 patent deals with events called “interrupts,” which are basically signals that are sent to the processor that interrupt its current operation. This occurs, for example, when a user moves an I/O device like a mouse, and the processor has to stop what it’s doing in order to move the cursor on the screen. See ‘495 patent at 2:15-29. When an interrupt occurs, the current PSW has to be saved “for later reestablishment when the interrupted program is later to be resumed.” *Id.* at 2:28-29. When that happens—i.e., when a particular PSW stops directing the processor and, instead, is saved in a “save area” for later use—that particular PSW is no longer *the* PSW but, instead, becomes a “saved PSW.” *Id.* at 5:48, 6:7, 11:34-35, 12:14, 13:17-18, 14:3-4. The “saved PSW” *cannot* be *the* PSW because there cannot be *two different instructions* that the processor is supposed to perform next. And that is the principal flaw in IBM’s proposed construction: IBM posits that the PSW is “a defined set of data,” but fails to say defined *by what*. Under IBM’s

¹⁹ As professor Patt explains in his declaration, when saved to memory this information becomes a *record* of what the PSW was at a prior point in time—but it is no longer the “program status word.” Patt Decl. ¶ 20.

construction a “saved program status word” in main memory *and* the current PSW in the processor’s PSW register are *both* the “program status word”—which cannot be right because the processor cannot simultaneously execute two different instructions. The intrinsic evidence expressly makes this point:

When an interruption occurs, the CPU places the current PSW in an assigned storage location, called the old-PSW location, for the particular class of interruption. The CPU fetches a new PSW from a second assigned storage location. *This new PSW determines the next program to be executed.* When it has finished processing the interruption, the interrupting program may reload the old PSW, *making it again the current PSW*, so that the interrupted program can continue.

390 POO at 2-3 (emphasis added). Because both parties agree that the PSW “indicates the next instruction to be executed” and “directs the processor in the execution of a program,” the “*saved* program status word” sitting in the storage location cannot be *the* “program status word,” because the intrinsic evidence establishes that only the *current* PSW (which the CPU has fetched and reloaded into the PSW register) can direct the processor.

The single example of a PSW directing a processor from somewhere other than in a specific PSW register located inside the processor that IBM was able to dig up is the old IBM System 360 Models 30 & 40. Smotherman ¶¶ 13-14. The Models 30 & 40, however, are from 1964, and are irrelevant to the construction of “program status word” in the patents-in-suit. The extrinsic evidence *from the time of the patents-in-suit* (i.e., over 30 years later) supports PSI’s construction:

- In a primer on “z/OS Concepts” (Ex. 48), IBM states: “The program status word (PSW) is a 128-bit data area *in the processor* that, along with a variety of *other types of registers* (control registers, timing registers, and prefix registers) provides details crucial to both the hardware and the software.” http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/topic/com.ibm.zconcepts.-doc/zconc_interrupts.html. The same document goes on to state: “Mainframe architecture provides registers to keep track of things. *The PSW*, for example, *is a register* used to contain information that is required for the execution of the currently active program.” *Id.*

- The *IBM Dictionary of Computing* defines the PSW not as “a defined set of data,” as IBM urges, but instead as “[a]n area in storage.” *IBM Dictionary of Computing* (Ex. 9) at 539.
- The *Computer Desktop Encyclopedia* (Ex. 14) defines “PSW” as “[a] **hardware register** that maintains the status of the program being executed.” <http://www.techweb.com/encyclopedia/defineterm.jhtml?term=PSW>.
- See also Patt Decl. ¶¶ 21-22.

In short, the intrinsic evidence uniformly supports PSI’s construction. The sole example to the contrary IBM could find is extrinsic evidence that dates to 1964, and is irrelevant to the proper construction of “program status word” in this lawsuit. PSI’s proposed construction should be adopted.

IV. THE INSTRUCTION/ARCHITECTURE PATENTS

A. The ‘495 “Resume Program” Patent

The mainframes at issue in this lawsuit can, like all modern computers, run more than one program at a time. Problems arise, however, when these programs have to share the limited resources available in the computer. Computer architects use a variety of techniques to address conflicting demands for those resources. Thus, for example, every program has an “operating authority” which specifies what the program can and cannot do. In general, there are at least two operating authorities: (1) “supervisor state,” in which the program can operate without authority constraints and (2) “problem state,” which “provides the logical and arithmetic capabilities necessary to solve the problems of a broad range of application programs” but in which a program’s abilities are otherwise restricted. ‘495 patent at 4:24-27. A computer’s operating system (*e.g.*, Microsoft Windows) typically operates in supervisor state, while an application running on the operating system (*e.g.*, Microsoft Word) typically runs in problem state.

Computer designers also have to deal with the fact that not all tasks have the same urgency. Some tasks, called “interrupts,” have an exceedingly high priority and must be handled

right away. When an operating program is disrupted so that the processor can handle such an “interrupt,” the processor must have some way to save its work and come back to it later. Specifically, the processor has to (1) save the contents of its registers, called the “program context,” to the memory; (2) load the information needed to handle the interrupt into those registers; and then, when the interrupt is complete, (3) reload the program context into its registers. *Id.* at 2:60-3:37.

But reloading the program context is tricky because some of the information that must be restored can only be restored in supervisor state. For example, one of the pieces of information that must be restored is the *authority* of the interrupted program. That information can only be restored in supervisor state. *Id.* at 3:55-59. And it is easy to see why: if programs operating in problem state could *change* their own authority by rewriting the contents of the register that defined that authority, the concept of a restricted-authority state would be illusory—just as the concept of a bedtime becomes illusory if you allow the ten-year-old to pick it for himself.

Notably, switching back and forth between problem state and supervisor state every time you want to restore a program context following an interrupt slows system performance. For that reason, IBM decided to build into its ISA a special instruction called “Resume Program,” which restores large parts of the program context *without* switching into supervisor state. *Id.* at 5:42-46. It is this instruction that is the subject of the ‘495 patent.

The parties have two sets of disputes with respect to the claim terms in the ‘495 patent: (1) how the terms “processor,” “instruction,” “program status word,” and “register” should be construed (discussed *supra* in Section III); and (2) what structure in the specification, if any, corresponds to certain means-plus-function limitations of claim 19.

1. All Of IBM’s Corresponding Structure Contentions Should Be Rejected As Improper Functional Claiming

Before delving into the four particular means-plus-function claim limitations at issue in the '495 patent, it is worth making a point that applies equally to IBM's identification of structure for all four of these limitations. It is black letter law that "[s]tructure disclosed in the specification is 'corresponding' structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim." *B. Braun Med. Inc. v. Abbott Labs.*, 124 F.3d 1419, 1424 (Fed.Cir.1997); *see also Medical Instrumentation & Diagnostics Corp. v. Elekta AB*, 344 F.3d 1205, 1211 (Fed.Cir.2003) ("If the specification is not clear as to the structure that the patentee intends to correspond to the claimed function, then the patentee has not paid the price but is attempting to claim in functional terms unbounded by any reference to structure in the specification."). The purpose behind this requirement of a clear identification and linkage of structure is to avoid functional claiming:

The point of the requirement that the patentee disclose particular structure in the specification and that the scope of the patent claims be limited to that structure and its equivalents is to avoid pure functional claiming.

Aristocrat Techs. Austl. PTY Ltd. v. Int'l Game Tech., 521 F.3d 1328, 1333 (Fed. Cir. 2008).

IBM's identification of corresponding structure for the four means-plus-function claim elements at issue in the '495 patent (and for nearly all of the other means-plus-function claim elements at issue in this lawsuit) contravenes the Federal Circuit's express holdings against functional claiming. IBM does not even pretend to limit itself to structures disclosed in the specification. Instead, for each of the four means-plus-function claim elements, IBM claims *all* "hardware, software or any suitable combination of the two" that is "configured" to perform the corresponding function:

Function	IBM's Claimed Corresponding Structure
"Decoding a program instruction..."	<i>Hardware, software, or any suitable combination of the two</i> (<u>see, e.g.</u> , Fig. 1, 102; 5:4-11; <u>or</u> 7:21-28) configured to decode an instruction from a program executing in said problem state specifying a storage location containing a saved program status ward (<u>see, e.g.</u> , Fig. 2A, 2B, 8:63-65, 9:5-15), and equivalents thereof.

“...executing said instruction...”	<i>Hardware, software, <u>or</u> any suitable combination</i> to execute an instruction (Fig. 1, 102; Col. 5:4-11; Col. 6:66-7:7, <u>or</u> Col. 7:21-4), and equivalents thereto.
“...accessing said save area...”	<i>Hardware, software, <u>or</u> any suitable combination of the two</i> (Fig. 1, 102; Col. 7:21- 28; <u>or</u> 5:4-11) that is configured to access the save area using the contents of the register specified by the program instruction (Fig. 2A; Fig. 2B, Fig. 3A; Fig. 6, steps 601, 602, 604, 606; Col. 8:63-65; Col. 9:5-22; Col. 10:42-45; Col. 10:46-50; Col. 10:54-59; <u>or</u> Col. 10:62- 66), and equivalents thereof
“...restoring said program status word...”	<i>Hardware, software, <u>or</u> any suitable combination of the two</i> (Fig. 1, 102; Col. 7:21- 28; <u>or</u> 5:4-11) that is configured to restore the program status word and the register from the saved program status word and saved register contents contained in the same area (Fig. 8, step 810; Fig. 6, steps 603, 605, <u>or</u> 607; Col. 9:58-64; Col. 10:50-53; Col. 10:59-61; <u>or</u> Col. 10:66-11:2), and equivalents thereof.

IBM Br. at 25, 27-29, 31 (emphasis added).

While one might think that IBM was relying on the citations in the parentheses as corresponding structure, IBM’s express use of “see, e.g.” and the disjunctive “or” makes clear that such is not the case. Instead, IBM is literally identifying “hardware, software, *or* any suitable combination of the two” as purported corresponding structure under 35 U.S.C. § 112 ¶ 6 for each of the four means-plus-function claim elements. For the “decoding...” function, for example, IBM identifies “hardware, software, or any suitable combination of the two” as the corresponding structure, and points to “Fig. 1, 102; 5:4-11; *or* 7:21-28” as *examples* (“see, e.g.,”) supporting its corresponding structure contention. For the “accessing...” function, IBM identifies the identical corresponding structure (“hardware, software, or any suitable combination of the two”) and the same three supporting examples (“Fig. 1, 102; Col. 7:21- 28; *or* 5:4-11”).

IBM’s corresponding structures for the ‘495 patent are an egregious example of functional claiming. This Court cannot be expected to seriously entertain the idea of construing “hardware, software, or any suitable combination of the two” as the corresponding structure for each of the four distinct functions at issue. In a case where *every* claim relates to computing, that is like saying the Court should construe “whatever decodes” as the corresponding structure for the “decoding” means, “whatever executes” as the corresponding structure for the “executes”